be2bill
Internet Payment

Strictement Confidentiel

Français | Version1.10 |17/09/2013

Documentation technique d'intégration

Table des matières

Introduction

Contacts

Be2bill, Rentabiliweb Europe SAS
55 rue Raspail
92300 Levallois-Perret
France
http://www.be2bill.com

Pour toute question, n'hésitez pas à nous contacter :

- Commercial : commercial@be2bill.com
- Technique : support@be2bill.com

Révisions de ce document

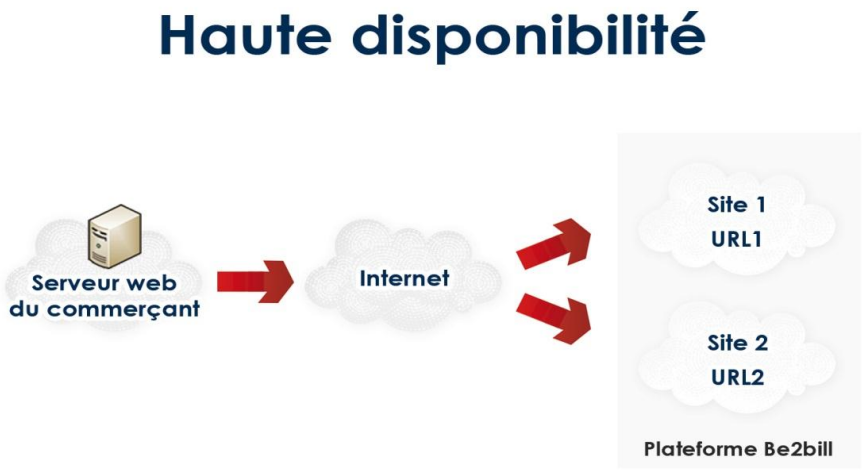
Nom	Chapitres / Pages	Version
Changement de la composition de la signature de requête (HASH)	- page 12 => Le paramètre HASH - page 50 => Signature HASH avec be2bill_signature	1.1
Ajout des paramètres HIDECLIENTEMAIL et HIDECARDFULLNAME dans le formulaire de transaction	- page 16 => Gestion des paramètres CARDFULLNAME et CLIENTEMAIL	1.1
Redéfinition des urls : **- https://$site/front/service/rest/process** **- https://$site/front/form/process**		1.1
Ajout de CARDTYPE, CARDFULLNAME et CARDVALIDITYDATE en url de notification	• page 33 URLS paramétrées	1.2
Ajout de CLIENTEMAIL au prototypage des appels		1.2
Application de l'habillage Be2bill		1.3
Retrait du mode serveur à serveur		1.4
Paiement récurrent avec CVV	• Page 21 Méthode «PAYMENT »	1.4
Champ CARDFULLNAME du formulaire remplace FIRSTNAME et LASTNAME	• Page 16 Paramètres POST du formulaire	1.4
Ajout de la section traitant de la sandbox	• Page 35 La « sandbox » Be2bill (mode bac à sable)	1.4
Ajout de 3DSECUREDISPLAYMODE	• Page 28 3DSECUREDISPLAYMODE	1.5
Retrait des instructions liées au crédit		1.6
Ajout du paiement en plusieurs fois	• Page 29 Option « paiement en plusieurs fois »	1.7
Ajout de DISPLAYCREATEALIAS	• Page 30 Gestion et utilisation des ALIAS	1.8
Nouvelle formule de calcul de hash, compatible paiement en plusieurs fois	• Page 50 Signature HASH avec be2bill_signature	1.9

Présentation

L'objectif de ce document est l'intégration technique de la solution Be2bill.
Pour vous aider, vous disposez de 3 documents vous permettant une intégration complète de la solution de paiement Be2bill:

- Document d'intégration générale
- Définition de tous les codes retours
- Définition de tous les champs paramètres utilisables

Gestion de la haute disponibilité



En plus des mécanismes de haute disponibilité transparents pour vous, déployés sur la plateforme Be2bill, nous vous offrons la possibilité de mettre en œuvre à votre convenance un mécanisme de bascule d'activité.
Pour réaliser des demandes de paiement, nous vous fournissons deux URLs au lieu d'une seule.
En temps normal vous devez utiliser l'URL principale uniquement.
Dans le cas où l'URL principale deviendrait indisponible, vous pourriez alors envoyer l'ensemble de vos demandes de paiement sur l'URL secondaire.
La liste des URLs vous parviendra ainsi que vos différents identifiants dans un document à part.

Attention : il ne s'agit pas d'un mécanisme d'équilibrage de charge, vous ne devez pas répartir vos demandes de paiement sur les deux URLs en régime nominal.

Exemples d'indisponibilité d'une URL :

- Aucune réponse au niveau réseau (IP, TCP)
- Codes d'erreurs HTTP (5XX, …)
- Codes d'erreur retournés par la plateforme (5001 et 5003)

Attention, les transactions d'éditions (ONECLICK, SUBSCRIPTION, REFUND, CAPTURE …) doivent être exécutées au minimum 5 minutes après la transaction de référence.

Exemple d'implémentation en PHP de la haute disponibilité

```php
<?php

#Url list provided by Be2bill
$url_list = array($url1, $url2);

function transaction($params)
{
    global $url_list;

    #Try on each declared url
    foreach ($url_list as $url)
    {
        #Prepare HTTPS request with curl
        $resource = curl_init();

        curl_setopt($resource, CURLOPT_URL, 'https://' . $url . '/front/service/rest/process');
        curl_setopt($resource, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($resource, CURLOPT_POST, true);
        curl_setopt($resource, CURLOPT_POSTFIELDS, http_build_query($params));

        #Send the request
        $serialized = curl_exec($resource);

        #Curl return something different from false
        if ($serialized !== false)
        {
            #Parse result
            $result = json_decode($serialized, true);

            #Handle result
            return $result;
        }
    }

    #Trigger error when no tryout works
    trigger_error('Unable to contact Be2bill, E_USER_ERROR);
}

#Init different parameters
$hash = '';
$AMOUNT = '';
# ...
```

```php
#Create an array with some parameters
$params = array(
    'method' => 'payment',
    'params' => array
    (
        'IDENTIFIER'        => BE2BILL_IDENTIFIER,
        'HASH'              => $hash,
        'AMOUNT'            => $AMOUNT,
        'CLIENTIDENT'       => $CLIENTIDENT,
        # ...
    )
);

$result = transaction($params);
```

Certificats SSL

Transport Layer Security (TLS) anciennement nommé Secure Sockets Layer (SSL) est un protocole de sécurisation des échanges sur Internet. Il fonctionne suivant un mode client-serveur et fournit les objectifs de sécurité suivants :

- l'authentification du serveur
- la confidentialité des données échangées
- l'intégrité des données échangées
- éventuellement l'authentification du client

A noter : les versions 1.0 et 2.0 du protocole sont obsolètes et réputées faillibles. Seules les versions SSL 3.0 / TLS 1.0 et supérieures doivent être utilisées.

Pour en savoir plus, vous pouvez commencer par consulter la page de Wikipédia :

- http://fr.wikipedia.org/wiki/Transport_Layer_Security

Certificat SSL sur votre serveur pour votre domaine

Si vous désirez intégrer des liens vers des éléments externes au formulaire de transactions (liens, images, CSS, scripts etc…) ou si vous désirez bénéficier d'une sécurité optimale lors des échanges Be2bill vers votre plateforme vous devez disposer d'un certificat SSL sur vos serveurs. Si vous n'avez pas de certificat SSL, vous devez en commander un auprès d'une autorité de certification (AC ou CA pour Certificate Authority).

C'est un prérequis plus que recommandé d'un point de vue sécurité. Sachez anticiper : selon les autorités de certification et le type de certificat que vous allez commander, la procédure complète peut prendre de 24h à plusieurs semaines. Nous vous invitons à consulter les offres de sites tels que :

- http://www.verisign.com/
- http://www.thawte.com/
- http://www.comodo.com/

Autorité de certification du certificat SSL Be2bill

La plateforme Be2bill utilise un certificat SSL qui inclut une validation étendue permettant notamment un chiffrement SSL supérieur à 256-bit.

99% des navigateurs web reconnaissent ces certificats et affiche alors une barre VERTE lorsque vous vous rendrez sur une adresse de la plateforme Be2bill, symbole de légitimité et de sécurisation.

Référence : http://www.thawte.fr/ssl/extended-validation-ssl-certificates/index.html

Si votre serveur web ne possède pas l'autorité de certification (AC ou CA) du certificat SSL de la plateforme Be2bill émis par Thawte, vous devez télécharger et ajouter ce fichier à la configuration de votre serveur.

Ce fichier est téléchargeable à partir de l'extranet Be2bill dans « Documentation Technique ».

Présentation des différentes opérations de la plateforme Be2bill

Paiement (payment)

La fonction « payment » est la fonction de base qui permet de débiter un porteur. Cette opération collecte directement l'argent.

Autorisation (authorization)

La fonction « authorization » permet de « bloquer » temporairement (7 jours) les fonds sur le compte bancaire d'un porteur. Cette fonctionnalité ne le débite pas.
Ce type d'opérations est surtout utilisé dans le monde des biens physiques (« le retail ») lorsque le marchand décide de débiter son client lors de l'envoi de la marchandise.

Capture

La fonction « capture » permet de collecter les fonds d'un porteur suite à une autorisation (fonction « authorization »). Cette capture peut intervenir dans un délai de 7 jours après l'autorisation.

Remboursement / Annulation (refund)

Cette fonction double se gère directement par le système :

- Remboursement : consiste à renvoyer des fonds déjà collecté à un porteur
- Annulation : consiste à ne pas envoyer en compensation une transaction de paiement

Communication avec la plateforme Be2bill

Le paramètre HASH

Chacune des requêtes Be2bill doit être signée avec l'algorithme de hachage SHA256.
La formule utilisée sur toute la plateforme est la suivante :

- Champ PASSWORD
- Ensemble des champs Be2bill composants la requête dans l'ordre alphabétique (a-z) sous la forme CHAMPS=valeur séparés par le champ PASSWORD
- Champ PASSWORD à la fin (encore une fois)

PASSWORD correspond bien entendu au mot de passe strictement confidentiel qui vous a été délivré à l'ouverture de votre compte.

Exemple de signature :
On désire transmettre une requête avec les champs suivants et « secret » pour mot de passe :

- AMOUNT = 10€
- DESCRIPTION = desc
- IDENTIFIER = 123
- CLIENTIDENT = johndoe

La formule de HASH sera donc la suivante :
SHA256(secretAMOUNT=1000secretCLIENTIDENT=johndoesecretDESCRIPTION=descsecretIDENTIFIER=123secret);

Attention:

- les paramètres non listés dans la documentation Be2bill ne doivent pas être inclus dans la signature (exemple : bouton submit, input hidden, csrf token etc…)
- le paramètre HASH ne doit pas lui-même être inclus dans la signature.
- dans les transactions de type serveur à serveur, OPERATIONTYPE doit être compris dans la signature, en minuscule et positionné dans l'ordre alphabétique par rapport aux autres champs.

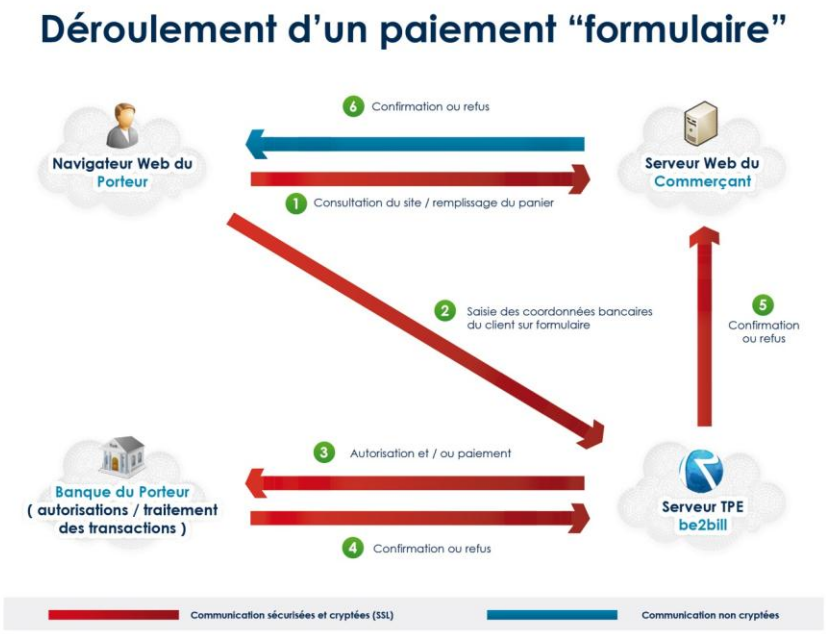
Remarque : utilisez la fonction de trie alphabétique de votre langage de programmation favori (en PHP vous pouvez utiliser ksort : http://php.net/manual/fr/function.ksort.php).

Reportez-vous à l'exemple d'implémentation en PHP pour plus d'informations : Signature HASH avec be2bill_signatureLe formulaire de transaction Be2bill

Présentation générale

L'intégralité des opérations nécessitant de transmettre les informations bancaires sensibles du porteur (numéro de carte, date de validité, cryptogramme etc…) se font par l'appel au formulaire de transaction.
Ce formulaire permet de déclencher des autorisations ou des paiements pourvu que le porteur fournisse les informations bancaires nécessaires.
Un paramètre (OPERATIONTYPE) permet de choisir quelle action déclenchera le formulaire (fonction « payment » ou « authorization »).

Le fonctionnement du formulaire de transaction est le suivant :



1/ Un utilisateur se connecte via son navigateur Internet sur un site marchand. A ce niveau la connexion n'est généralement pas cryptée.

2/ Lorsque l'utilisateur décide de payer sa commande il est dirigé vers une page de saisie de coordonnées bancaires. A ce stade, les échanges avec la plateforme Be2bill sont cryptés.

3/ Une fois les coordonnées de paiement saisies par l'utilisateur, le site marchand fait un appel à la méthode « payment » de notre service Web. C'est ce dernier qui va se charger de transmettre la transaction au réseau bancaire.

4/ Si les données sont valides, notre plateforme demande l'autorisation auprès du réseau de cartes bancaires et réalise ensuite le prélèvement, toujours à l'aide d'une connexion cryptée.

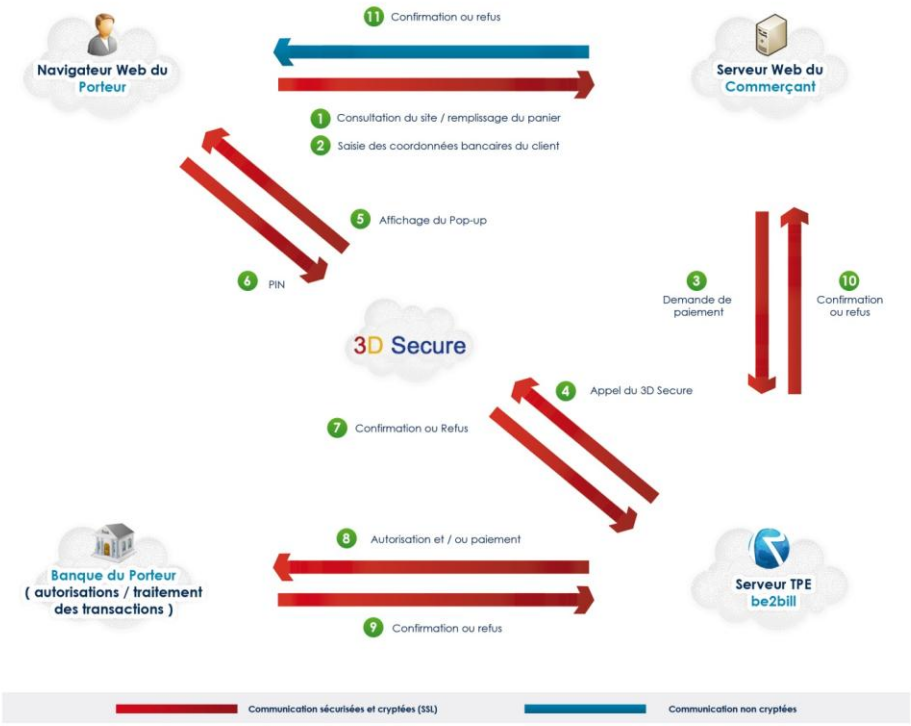
5/ Le réseau bancaire indique à notre plateforme si le paiement est accepté ou refusé. (Connexion cryptée).

6/ Notre plateforme confirme au site marchand le succès ou l'échec du paiement auprès du réseau bancaire (Connexion cryptée).

7/ Le marchand confirme ensuite à son client le succès ou non du paiement.

Le fonctionnement du formulaire 3DSecure de transaction est le suivant :

Déroulement d'un paiement 3D Secure



1/ Un utilisateur se connecte via son navigateur Internet sur un site marchand. A ce niveau la connexion n'est généralement pas cryptée.

2/ Lorsque l'utilisateur décide de payer sa commande il est dirigé vers une page de saisie de coordonnées bancaires. (Connexion cryptée).

3/ Une fois les coordonnées de paiement saisies par l'utilisateur, le site marchand fait un appel à la méthode « payment » (ou « authorization ») de notre service Web. C'est ce dernier qui va se charger de transmettre la transaction aux intervenants 3DSecure et au réseau bancaire.
4/ La plateforme fait un appel au réseau 3DSecure afin de pouvoir authentifier la transaction.

5/ Un formulaire de la banque du porteur s'affiche sur le navigateur du porteur (connexion cryptée).
6/ Le porteur rentre son PIN (code reçu par SMS, date de naissance, …)

7/ Le réseau 3DSecure valide ou non la transaction suite à l'authentification du porteur (connexion cryptée).

8/ Notre plateforme va décider de poursuivre ou pas la transaction en fonction du résultat de l'authentification 3DSecure.

9/ Le réseau bancaire indique à notre plateforme si le paiement est accepté ou refusé. (Connexion cryptée).

10/ Notre plateforme confirme au site marchand le succès ou l'échec du paiement auprès du réseau bancaire (Connexion cryptée).

11/ Le marchand confirme ensuite à son client le succès ou non de la transaction.

Paramètres POST du formulaire

Pour afficher un formulaire de transaction, il suffit de faire une redirection vers https://$url/front/form/process en spécifiant les paramètres POST suivants :

Nom	Obligatoire
IDENTIFIER	Oui
HASH	Oui
OPERATIONTYPE (PAYMENT OU AUTHORIZATION)	Oui
CLIENTIDENT	Oui
DESCRIPTION	Oui
ORDERID	Oui
VERSION	Oui
AMOUNT	Oui
CARDTYPE	Non
CLIENTEMAIL	Non
CARDFULLNAME	Non
LANGUAGE	Non
EXTRADATA	Non
CLIENTDOB	Non
CLIENTADDRESS	Non
CREATEALIAS	Non
3DSECURE	Non
3DSECUREDISPLAYMODE	Non
USETEMPLATE	Non
HIDECLIENTEMAIL	Non
HIDECARDFULLNAME	Non

Gestion des paramètres CARDFULLNAME et CLIENTEMAIL

Lors de l'utilisation du formulaire vous pouvez décider de pré-remplir les champs CARDFULLNAME et CLIENTEMAIL avec des données issues de votre base de données. Pour cela, il suffit de spécifier dans la requête d'appel au formulaire ces différentes valeurs. Ainsi, les porteurs n'auront qu'à confirmer ces informations sur la page de formulaire.

REMARQUE : Vous pouvez masquer sur le formulaire de paiement la demande de l'E-mail et du nom du porteur de carte :

Si vous ne voulez pas que ces paramètres soient éditables ou affichés sur le formulaire vous pouvez fournir les paramètres HIDECLIENTEMAIL et HIDECARDFULLNAME mais dans ce cas, les champs CLIENTEMAIL et CARDFULLNAME deviennent obligatoire à l'initialisation du formulaire.

Il est toutefois préférable que le porteur valide ces informations, en effet dans certains cas, les informations du porteur (noms et prénoms présents sur la carte) diffèrent de celles liées au compte utilisateur présent dans votre base de données.

Notification du résultat du formulaire

Il est possible de configurer dans l'extranet Be2bill un URL vers votre boutique vous permettant de recevoir le résultat de chaque transaction.

Cet URL sert à mettre à jour votre back office client en fonction du résultat de la transaction. C'est suite à l'envoie d'une notification de la plateforme Be2bill sur cet URL qu'il vous faudra :

- Envoyer un Email au porteur de confirmation au porteur (commande acceptée, refusée …)
- Mettre à jour votre base de données
- Activer le service ou expédier la commande si la transaction a réussi
- …

Le statut final de la transaction s'obtient par la variable EXECCODE transmise dans la requête.
Vous pouvez vous reporter au chapitre « Paramètres de configuration » en page 26.

Personnalisation du formulaire

Le formulaire de paiement est personnalisable à souhait selon votre charte graphique.
Pour cela, il vous faut fournir sur l'extranet Be2bill un URL modèle pour le formulaire (liste des comptes / configurer).
L' URL modèle doit référencer une page hébergée sur votre serveur qui contiendra le modèle (ou gabarit) à appliquer autour du formulaire.

Vous pouvez bien entendu utiliser une simple page HTML statique ou un langage dynamique (PHP, ASP etc…) pour générer le HTML qui convient.

Vous pouvez vous reporter au chapitre « URLS paramétrées ».

La page modèle présente sur votre serveur devra contenir une balise %PLACEHOLDER% dans le code source. La variable %PLACEHOLDER définit ainsi l'emplacement où le formulaire de paiement doit être positionné.

Vous pouvez fournir un modèle différent pour les terminaux WEB et un modèle différent pour les terminaux MOBILE.

Si le modèle MOBILE n'est pas défini, le modèle WEB sera utilisé par défaut.
Vous pouvez forcer le mode WEB ou MOBILE via le paramètre USETEMPLATE.

Voici un exemple de contenu que le modèle doit retourner :

```
<html>
   <head>
      <title>Title</title>
      <link type="text/css" href="https://my_site_with_https.com/myfile.css" media="all" />
   </head>
   <body>
      <p>ORDER ID: <?php echo $_POST[ORDERID] ?></p>
      <p>The formular:</p>
      %PLACEHOLDER%
   </body>
</html>
```

Attention! Si vous décidez de référencer des scripts, images ou d'autres éléments non compris dans le code source, vous devrez impérativement fournir le chemin absolu vers un url HTTPS (HTTP+ SSL) uniquement.

Pensez donc à vous munir d'un certificat SSL.

Redirections du formulaire

Il est possible de configurer des URL de redirection propres au formulaire :

- URL de redirection après une transaction
- URL de redirection lors de l'annulation d'une transaction

L'URL de redirection en fin de transaction vous permet de définir où rediriger le porteur une fois la transaction formulaire terminée.

L'URL de redirection pour annulation vous permet de définir où rediriger le porteur si ce dernier clic sur le bouton « Annuler ».

Si vous ne configurez pas l'URL de redirection en cas d'annulation, le bouton « annuler » ne sera pas présent sur le formulaire.
Vous pouvez vous reporter au chapitre « Paramètres de configuration (extranet Be2bill) ».

Le service web « REST – LIKE »

La plateforme Be2bill peut fonctionner par requêtes POST HTTPS (HTTP + SSL) de serveur à serveur.

L'ensemble des opérations ne nécessitant pas de spécifier les informations bancaire du porteur (numéro de carte etc…) peuvent s'effectuer en mode « serveur à serveur » via le protocole REST.

Une requête REST Be2bill se compose

- d'un URL (https://$site/front/service/rest/process)
- des paramètres décrivant l'action souhaitée en POST
- du retour de la requête au format JSON

Paramètres REST

Voici les paramètres à transmettre en POST à chaque appel au service REST

Nom	Type	Description	Obligatoire
method	Champs texte	L'action à déclencher	Oui
params	Tableau associatif	Un tableau clés/valeurs contenant les différents paramètres liés à l'action	Oui

Paramètres liés à l'action (« params »)

Dans le champ « params », il faudra systématiquement fournir les éléments suivants :

Nom	Obligatoire
IDENTIFIER	Oui
HASH	Oui
VERSION	Oui

A ces paramètres obligatoires s'ajouteront d'autres paramètres en fonction de l'action désirée.

Paramètres de retour

Le retour de l'appel au service web se constitue toujours d'un tableau sérialisé au format JSON.

Le tableau contiendra en général les paramètres :

Nom	Obligatoire
OPERATIONTYPE	Oui
EXECCODE	Oui
MESSAGE	Oui
TRANSACTIONID	Non
DESCRIPTOR	Non

Le champ TRANSACTIONID correspond à l'identifiant technique unique qu'il vous faudra obligatoirement conserver dans votre système d'information.

Les demandes de support et de nombreuses requêtes techniques nécessitent la transmission de ce paramètre.

Le paramètre TRANSACTIONID est présent obligatoirement en cas de réussite d'une transaction et dans la majorité des cas de refus. Certains cas d'erreurs particuliers peuvent entrainer l'absence de ce champ (exemple : champs requis manquant, mauvais identifiant de compte …)
A ces champs s'ajouteront d'autres informations liées à l'action demandée.

Voici un exemple de retour typique d'appel au service web :

{"OPERATIONTYPE":"PAYMENT","EXECCODE":"0000","MESSAGE":"The transaction has been accepted","TRANSACTIONID":"AB123456","DESCRIPTOR":"site.com"}

Intégration PayPal / Buyster

L'utilisation des services de paiement PayPal et Buyster recourt au service web « REST – LIKE », et nécessite la création de comptes spécifiques Be2bill.

En cas de réception du code retour EXECCODE=0002, la redirection vers les formulaires PayPal ou Buyster se fait en affichant le contenu du paramètre REDIRECTHTML, après l'avoir décodé (base64).

Après validation de la transaction, une ultime redirection est effectuée vers l'URL indiquée dans la configuration de votre espace Extranet Be2bill.

Le résultat de la transaction vous sera transmis par notification HTTP. (voir la section URLS paramétrées)

Méthode «PAYMENT »

La méthode « payment » sert à déclencher des transactions de paiement en mode « serveur à serveur » dans le cas de transactions récurrentes (abonnement ou 1click).

Pour ces transactions vous pouvez définir de passer ou pas le CARDCVV. Si le CARDCVV est fourni, il sera à nouveau validé au moment de l'autorisation.

Vous pouvez vous reporter au chapitre « Gestion et utilisation des ALIAS ».

Paramètres attendus

Nom	Obligatoire
IDENTIFIER	Oui
HASH	Oui
OPERATIONTYPE	Oui
ALIAS	Oui
ALIASMODE	Oui
DESCRIPTION	Oui
ORDERID	Oui
VERSION	Oui
AMOUNT	Oui
CLIENTREFERRER	Oui
CLIENTUSERAGENT	Oui
CLIENTIP	Oui
CLIENTEMAIL	Oui
CLIENTIDENT	Oui
CARDCVV	Non
EXTRADATA	Non
CLIENTDOB	Non
CLIENTADDRESS	Non
3DSECURE	Non
3DSECUREDISPLAYMODE	Non
LANGUAGE	Non

Retour de l'appel

Nom	Commentaire	Obligatoire
OPERATIONTYPE		Oui
EXECCODE		Oui
MESSAGE		Oui
TRANSACTIONID		Non
DESCRIPTOR		Non
3DSECUREHTML	Champs base64 encodé à afficher au porteur pour déclencher l'authentification 3DSECURE	Non

Méthode «AUTHORIZATION »

La méthode « authorization» sert à déclencher des transactions d'autorisation dans le cas de transactions récurrentes (abonnement ou 1click). Ces transactions devront ensuite être confirmées manuellement via l'opération «capture » afin de les inscrire à la prochaine télécollecte.

Pour ces transactions vous pouvez définir de passer ou pas le CARDCVV. Si le CARDCVV est fourni, il sera à nouveau validé au moment de l'autorisation.

Vous pouvez vous reporter au chapitre « Gestion et utilisation des ALIAS ».

Paramètres attendus

Nom	Obligatoire
IDENTIFIER	Oui
HASH	Oui
OPERATIONTYPE	Oui
ALIAS	Oui
ALIASMODE	Oui
DESCRIPTION	Oui
ORDERID	Oui
VERSION	Oui
AMOUNT	Oui
CLIENTREFERRER	Oui
CLIENTUSERAGENT	Oui
CLIENTIP	Oui
CLIENTEMAIL	Oui
CLIENTIDENT	Oui
CARDCVV	Non
EXTRADATA	Non
CLIENTDOB	Non
CLIENTADDRESS	Non
3DSECURE	Non
3DSECUREDISPLAYMODE	Non
LANGUAGE	Non

Retour de l'appel

Nom	Description	Obligatoire
OPERATIONTYPE		Oui
EXECCODE		Oui
MESSAGE		Oui
TRANSACTIONID		Non
DESCRIPTOR		Non
3DSECUREHTML	Champs base64 encodé à afficher au porteur pour déclencher l'authentification 3DSECURE	Non

Méthode « CAPTURE »

La méthode capture s'utilise pour « confirmer » une demande d'autorisation faite au préalable (méthode « authorization »).

Vous pouvez ainsi implémenter des fonctionnalités du type « paiement à l'expédition » etc…

La durée maximum entre une capture et son autorisation est de 7 jours. Passé ce délai, la capture a de fortes chances d'être refusée par le réseau bancaire.
Une capture doit impérativement référencer une transaction réussie de type « authorization » sinon elle sera refusée.

Vous ne pouvez pas capturer plusieurs fois la même autorisation (Une autorisation = 1 capture).

Paramètres attendus

Nom	Description	Obligatoire
IDENTIFIER		Oui
HASH		Oui
OPERATIONTYPE		Oui
TRANSACTIONID	L'identifiant de l'autorisation réussie que vous souhaitez confirmer.	Oui
DESCRIPTION		Oui
ORDERID		Oui
VERSION		Oui
AMOUNT	Vous pouvez décider de ne prélever qu'une partie du montant défini lors de l'appel à « authorization ». AMOUNT_CAPTURE <= AMOUNT_AUTH	Non
EXTRADATA		Non

Retour de l'appel

Nom	Obligatoire
OPERATIONTYPE	Oui
EXECCODE	Oui
MESSAGE	Oui
TRANSACTIONID	Non
DESCRIPTOR	Non

Méthode « REFUND »

La méthode « refund » permet de rembourser ou d'annuler une transaction réussie.
Si la transaction à rembourser/annuler a été faite dans un délai suffisamment cours (n'a pas été mise en télécollecte), l'appel à « refund » entrainera une annulation de paiement (par opposition à un remboursement). Ainsi, le porteur ne verra aucune trace du paiement initial sur son relevé bancaire.

TRANSACTIONID doit impérativement référencer une transaction de type « payment » ou « capture » réussie.

Paramètres attendus

Nom	Description	Obligatoire
IDENTIFIER		Oui
HASH		Oui
OPERATIONTYPE		Oui
TRANSACTIONID	L'identifiant de la transaction réussie que vous souhaitez rembourser ou annuler.	Oui
DESCRIPTION		Oui
ORDERID		Oui
VERSION		Oui
EXTRADATA		Non

Retour de l'appel

Nom	Description	Obligatoire
OPERATIONTYPE		Oui
EXECCODE		Oui
MESSAGE		Oui
TRANSACTIONID		Non
DESCRIPTOR		Non

Options des opérations de la plateforme Be2bill

Option « 3DSECURE »

Sur un compte supportant la technologie 3DSecure, il suffit d'adjoindre aux transactions de type « payment » ou « authorization » le paramètre 3DSECURE = YES pour déclencher le processus d'authentification et bénéficier de la sécurisation 3DSecure.

Mode serveur à serveur

En mode serveur à serveur, le retour d'une transaction 3DSECURE contiendra un EXECCODE particulier et un champ supplémentaire :

Nom	Description
EXECCODE	Si carte enrôlée : EXECCODE = 0001
MESSAGE	Si carte enrôlée : « Identification requested »
3DSECUREHTML	Chaine base64 contenant le code HTML de redirection pour le porteur

Le champ 3DSECUREHTML contient une chaine HTML encodée en base64 qu'il vous faudra donc décoder et afficher au porteur pour le rediriger vers la page d'authentification de sa banque.

La transaction est ainsi figée jusqu'à l'authentification ou l'abandon du porteur sur le formulaire de la banque porteur.

Pour connaitre le résultat d'une transaction 3DSecure en mode serveur à serveur, vous devez :

- Configurer un URL de notification de transaction dans l'interface Be2bill
- Configurer un URL de redirection dans l'interface Be2bill

3DSECUREDISPLAYMODE

Vous pouvez choisir avec le paramètre 3DSECUREDISPLAYMODE la façon dont le formulaire d'authentification 3DSECURE sera présenté au porteur.

Nom	Description
MAIN	La page est affichée dans la page/IFRAME courante
POPUP	La page est affichée dans une fenêtre du navigateur
TOP	La page est affichée dans la page de plus haut niveau (rafraichissant même les IFRAME)

Remarque :

Attention, le comportement POPUP dépend fortement de la capacité du navigateur du porteur à afficher ou non les POPUP. Ce mode d'affichage est fortement déconseillé.

Option « paiement en plusieurs fois »

Il est possible de fractionner un paiement en plusieurs échéances, par exemple dans le cadre de la mise en place d'une facilité de paiement.
Il suffit de remplacer le paramètre AMOUNT par un tableau associatif AMOUNTS dont les clés / valeurs correspondent respectivement à la date de chaque échéance et à son montant :

Si nous sommes le 17-01-2013 et que nous voulons débiter 334€ aujourd'hui, 333€ le 1 mars, 333€ le 1 avril :

AMOUNTS[2013-01-17]=33400&AMOUNTS[2013-03-01]=33300&AMOUNTS[2013-04-01]=33300

Cette requête doit respecter quelques conditions :

- La première échéance doit être programmée le jour de la requête et sera exécutée immédiatement.
- L'intervalle entre la première et la dernière échéance ne doit pas excéder 90 jours.
- La dernière échéance doit tomber avant la date d'expiration de la carte utilisée.

Remarques :

En cas de refus de la première transaction uniquement, l'ensemble la requête est annulée.
Les dates sont attendues au format UTC.

Gestion et utilisation des ALIAS

Il est possible de pouvoir débiter un porteur sans avoir à demander systématiquement la ressaisie des informations sensibles (numéro de carte, cryptogramme etc…)

Cette fonctionnalité permet d'implémenter entre autre des fonctionnalités d'abonnement (« subscription ») ou de paiement en un clic (« oneclick ») de façon sécurisée (en accord avec la norme de sécurité PCI-DSS).

Pour cela il vous faudra soit spécifier le paramètre CREATEALIAS sur la première transaction du porteur :

Nom		Description
CREATEALIAS	YES ou NO	Demande la création d'un alias du porteur

Soit laisser le choix à vos clients d'enregistrer ses coordonnées bancaires pour ne pas avoir à les ressaisir lors d'un futur achat :

Nom		Description
DISPLAYCREATEALIAS	YES ou NO	Affiche une *checkbox* dans le formulaire de paiement pour sauvegarder les données de carte du porteur

En cas de succès (uniquement) le retour de la transaction contiendra un nouveau champ :

Nom	Description	Obligatoire
ALIAS	Identifiant du porteur (à conserver)	Oui si succès

Ce champ à conserver dans votre base de données sert à référencer la carte du porteur.

Ainsi, lors des transactions ultérieures, il vous suffira d'utiliser l'ALIAS obtenu dans le retour de la transaction initiale, à la place des informations bancaires du porteur (CARDCODE, CARDCVV …)
Cet ALIAS de carte pourra être utilisé pour toutes les transactions ultérieures du porteur de type

- « authorization »
- « payment »

Pour utiliser un alias dans une transaction il suffit d'ajouter les paramètres :

Nom	Description
ALIAS	L'alias lié au porteur que l'on souhaite débiter
ALIASMODE	Le mode d'utilisation de l'alias (ONECLICK ou SUBSCRIPTION)

Le champ ALIASMODE est un complément obligatoire du paramètre ALIAS. Il sert à indiquer le contexte métier d'utilisation de l'ALIAS

ALIASMODE = SUBSCRIPTION

Les transactions d'abonnement (option SUBSCRIPTION) correspondent à des transactions récurrentes dans le temps.

Les transactions sont déclenchées par le site marchand (vous) suite à une seule acceptation du porteur lors de la création de son abonnement.

Le consentement du porteur ne sera pas revalidé à chaque transaction.
Exemple d'utilisation :

- Cotisations tous les 3 mois pour l'accès PREMIUM à votre site

Dans un tel cas d'utilisation, vous effectuerez un appel à la fonction « payment » au service web tous les 3 mois en indiquant le montant et l'alias à débiter.

ALIASMODE = ONECLICK

Les transactions de type 1clic (option ONECLICK) correspondent à des transactions à l'initiative du porteur.

La fonctionnalité permet ainsi d'économiser la demande de saisie des informations bancaires (carte, date, cvv …) du porteur à chaque transaction, augmentant ainsi l'aspect « compulsif » du processus.

Les transactions 1clic doivent être initiées avec le consentement systématique du porteur.

Exemple d'utilisation :

- Rechargement rapide en jetons sur le compte de l'utilisateur

- Achat d'une musique ou application sur le téléphone via un système de « market place »

Remarque : Pensez à proposer à vos utilisateurs de pouvoir renseigner une nouvelle carte (cas de la carte expirée perdue ou volée).

Paramètres de configuration (extranet Be2bill)

Adresses IP autorisées

Dans le cas de transactions serveur à serveur, il est indispensable de renseigner les adresses IP de votre plateforme qui seront amenées à converser avec la plateforme Be2bill.
Ceci peut s'effectuer dans la page de configuration des comptes de l'extranet Be2bill.
Vous pouvez configurer au maximum 255 adresses IP différentes par compte.

URLS paramétrées

Il est possible dans la gestion de comptes de l'extranet Be2bill de paramétrer différents URL de notification, redirection et de lien vers des pages modèles.
Lors de l'utilisation de ces différents URLs des variables seront transmises en GET et en POST lorsque cela est possible :

URL	Méthode	Description
Notification de transactions	GET et POST	Appelée automatiquement à l'issue de chaque transaction pour vous transmettre le résultat. C'est cet appel qui vous fera valider ou pas une commande.
Notification d'impayés	GET et POST	Appelée à chaque nouvel impayé pour vous permettre de les saisir automatiquement dans votre base de données.
Modèle formulaire web	GET et POST	Définit le modèle (gabarit) pour y incruster le formulaire de transaction.
Modèle formulaire mobile	GET et POST	Définit le modèle (gabarit) pour y incruster le formulaire de transaction pour mobile.
Redirection après transaction	GET	Définit la page où rediriger le porteur après une transaction formulaire ou 3DSECURE
Redirection après annulation	GET	Définit le lien de retour vers votre site sur le formulaire de transaction.

Voici les différents paramètres transmis sur ces URLs :

Nom	Description	Exemple
IDENTIFIER		
HASH		
OPERATIONTYPE		
TRANSACTIONID		
CLIENTIDENT		
CLIENTEMAIL		
ORDERID		
AMOUNT		
VERSION		
LANGUAGE		
EXTRADATA		
CARDCODE	4 derniers chiffres de la carte uniquement	XXXXXXXXXXXX1234
CARDVALIDITYDATE		04-13
CARDFULLNAME		
CARDTYPE		
CARDCOUNTRY	Pays dans lequel la carte de paiement a été délivrée	FR
EXECCODE		0000
MESSAGE		The transaction has been accepted
DESCRIPTOR		
CURRENCY	Euros	EUR
ALIAS		A123456

Important : pour que les appels aux URLs de notifications soient considérés comme réussis vous devez impérativement afficher la chaine OK en résultat de la requête.

La « sandbox » Be2bill (mode bac à sable)

La plateforme Be2bill dispose d'un environnement dédié aux tests et à l'intégration : la « sandbox »

Les transactions envoyées sur cet environnement ne génèrent ni réelle autorisation bancaire, ni télécollectes : elles sont entièrement simulées.

Important : Ne jamais envoyer de données de production ou d'informations porteur réelles sur cette plateforme !

Le service de « sandbox » fonctionne de la même manière que le service de production si ce n'est qu'il reste joignable par un URL différent de la production et qu'il nécessite un couple IDENTIFIER, PASSWORD différent de votre compte de production.

Dans la documentation traitant des codes retour vous trouverez des PAN permettant de déclencher des retours particuliers de refus bancaire (scénarios de refus basés sur le PAN).

L'URL ($url) de la sandbox est : https://secure-test.be2bill.com/

Exemple d'implémentation simple en langage PHP5

Les exemples ci-dessus sont donnés uniquement à titre illustratif. Ils ne sont absolument pas exhaustifs en termes de gestion de la haute disponibilité et de sécurité.

Formulaire de paiement

```
<?php

define(BE2BILL_IDENTIFIER, 'my_id');
define(BE2BILL_PASSWORD, 'this_is_a_secret');

$AMOUNT = intval($order['amount'] * 100);
$AMOUNTS = array(
    $
) intval($order['amount'] * 100);
$CLIENTIDENT = $user['id'] . '_' . $user['login'];
$CLIENTADDRESS = $user['address1'] . $user['address2'] . $user['cp'] .
$user['city'] . $user['country'];

$hash = be2bill_signature(…) ;

?>

...
<form method="POST" action="https://$site/front/form/process">
    <!-- mandatory fields -->
    <input type="hidden" name="IDENTIFIER" value="<?php echo
BE2BILL_IDENTIFIER ?>" />
    <input type="hidden" name="HASH" value="<?php echo $hash ?>" />
    <input type="hidden" name="OPERATIONTYPE" value="payment" />
    <input type="hidden" name="CLIENTIDENT" value="<?php echo $CLIENTIDENT
?>" />
    <input type="hidden" name="DESCRIPTION" value="<?php echo
$order['description'] ?>" />
    <input type="hidden" name="ORDERID" value="<?php echo $order['id'] ?>" />
          <input type="hidden" name="AMOUNT" value="<?echo $AMOUNT ?>" />
    <input type="hidden" name="VERSION" value="2.0" />

    <!-- Some optionnal fields -->
    <input type="hidden" name="CLIENTADDRESS" value="<?php echo
$CLIENTADDRESS ?>" />
    <input type="hidden" name="CLIENTEMAIL" value="<?php echo $user['email']
?>" />
```

```
    <input type="submit" name="ok" value="Buy with Be2bill " />
</form>
...
```

Formulaire de paiement 3DSecure

```
<?php

define(BE2BILL_IDENTIFIER, 'my_id');
define(BE2BILL_PASSWORD, 'this_is_a_secret');

$AMOUNT = intval($order['amount'] * 100);
$CLIENTIDENT = $user['id'] . '_' . $user['login'];
$CLIENTADDRESS = $user['address1'] . $user['address2'] . $user['cp'] .
$user['city'] . $user['country'];

$hash = be2bill_signature(…) ;

?>

...
<form method="POST" action="https://$site/front/form/process">
    <!-- mandatory fields -->
    <input type="hidden" name="IDENTIFIER" value="<?php echo
BE2BILL_IDENTIFIER ?>" />
    <input type="hidden" name="HASH" value="<?php echo $hash ?>" />
    <input type="hidden" name="OPERATIONTYPE" value="payment" />
    <input type="hidden" name="3DSECURE" value="YES" />
    <input type="hidden" name="CLIENTIDENT" value="<?php echo $CLIENTIDENT
?>" />
    <input type="hidden" name="DESCRIPTION" value="<?php echo
$order['description'] ?>" />
    <input type="hidden" name="ORDERID" value="<?php echo $order['id'] ?>" />
          <input type="hidden" name="AMOUNT" value="<?echo $AMOUNT ?>" />
    <input type="hidden" name="VERSION" value="2.0" />

    <!-- Some optional fields -->
    <input type="hidden" name="CLIENTADDRESS" value="<?php echo
$CLIENTADDRESS ?>" />
    <input type="hidden" name="CLIENTEMAIL" value="<?php echo $user['email']
?>" />

    <input type="submit" name="ok" value="Buy with Be2bill " />
</form>
...
```

Formulaire de paiement en plusieurs fois

```
<?php

define(BE2BILL_IDENTIFIER, 'my_id');
define(BE2BILL_PASSWORD, 'this_is_a_secret');

$AMOUNTS = array(
    $schedule1 => intval($order['amounts1'] * 100),
    $schedule2 => intval($order['amounts2'] * 100),
    $schedule3 => intval($order['amounts3'] * 100),
)
$CLIENTIDENT = $user['id'] . '_' . $user['login'];
$CLIENTADDRESS = $user['address1'] . $user['address2'] . $user['cp'] .
$user['city'] . $user['country'];

$hash = be2bill_signature(…) ;

?>

...
<form method="POST" action="https://$site/front/form/process">
    <!-- mandatory fields -->
    <input type="hidden" name="IDENTIFIER" value="<?php echo
BE2BILL_IDENTIFIER ?>" />
    <input type="hidden" name="HASH" value="<?php echo $hash ?>" />
    <input type="hidden" name="OPERATIONTYPE" value="payment" />
<?php foreach( $AMOUNTS as $schedule => $amount ) : ?>
    <input type="hidden" name="AMOUNTS[<?php echo $schedule ?>]"
value="<?php echo $amount ?>" />
<?php endforeach; ?>
    <input type="hidden" name="CLIENTIDENT" value="<?php echo $CLIENTIDENT
?>" />
    <input type="hidden" name="DESCRIPTION" value="<?php echo
$order['description'] ?>" />
    <input type="hidden" name="ORDERID" value="<?php echo $order['id'] ?>" />
    <input type="hidden" name="VERSION" value="2.0" />

    <!-- Some optional fields -->
    <input type="hidden" name="CLIENTADDRESS" value="<?php echo
$CLIENTADDRESS ?>" />
    <input type="hidden" name="CLIENTEMAIL" value="<?php echo $user['email']
?>" />

    <input type="submit" name="ok" value="Buy with Be2bill " />
</form>
...
```

Formulaire d'autorisation

```
<?php

define(BE2BILL_IDENTIFIER, 'my_id');
define(BE2BILL_PASSWORD, 'this_is_a_secret');

$AMOUNT = intval($order['amount'] * 100);
$CLIENTIDENT = $user['id'] . '_' . $user['login'];
$CLIENTADDRESS = $user['address1'] . $user['address2'] . $user['cp'] .
$user['city'] . $user['country'];

$hash = be2bill_signature(…) ;

?>

...
<form method="POST" action="https://$site/front/form/process">
    <!-- mandatory fields -->
    <input type="hidden" name="IDENTIFIER" value="<?php echo
BE2BILL_IDENTIFIER ?>" />
    <input type="hidden" name="HASH" value="<?php echo $hash ?>" />
    <input type="hidden" name="OPERATIONTYPE" value="authorization" />
    <input type="hidden" name="CLIENTIDENT" value="<?php echo $CLIENTIDENT
?>" />
    <input type="hidden" name="DESCRIPTION" value="<?php echo
$order['description'] ?>" />
    <input type="hidden" name="ORDERID" value="<?php echo $order['id'] ?>" />
        <input type="hidden" name="AMOUNT" value="<?echo $AMOUNT ?>" />
    <input type="hidden" name="VERSION" value="2.0" />

    <!-- Some optionnal fields -->
    <input type="hidden" name="CLIENTADDRESS" value="<?php echo
$CLIENTADDRESS ?>" />
    <input type="hidden" name="CLIENTEMAIL" value="<?php echo $user['email']
?>" />

    <input type="submit" name="ok" value="Buy with Be2bill " />
</form>
...
```

Formulaire d'autorisation 3DSecure

```
<?php

define(BE2BILL_IDENTIFIER, 'my_id');
define(BE2BILL_PASSWORD, 'this_is_a_secret');

$AMOUNT = intval($order['amount'] * 100);
$CLIENTIDENT = $user['id'] . '_' . $user['login'];
$CLIENTADDRESS = $user['address1'] . $user['address2'] . $user['cp'] .
$user['city'] . $user['country'];

$hash = Be2bill_signature(…) ;

?>

...
<form method="POST" action="https://$site/front/form/process">
    <!-- mandatory fields -->
    <input type="hidden" name="IDENTIFIER" value="<?php echo
BE2BILL_IDENTIFIER ?>" />
    <input type="hidden" name="HASH" value="<?php echo $hash ?>" />
    <input type="hidden" name="OPERATIONTYPE" value="authorization" />
    <input type="hidden" name="3DSECURE" value="YES" />
    <input type="hidden" name="CLIENTIDENT" value="<?php echo $CLIENTIDENT
?>" />
    <input type="hidden" name="DESCRIPTION" value="<?php echo
$order['description'] ?>" />
    <input type="hidden" name="ORDERID" value="<?php echo $order['id'] ?>" />
          <input type="hidden" name="AMOUNT" value="<?echo $AMOUNT ?>" />
    <input type="hidden" name="VERSION" value="2.0" />

    <!-- Some optionnal fields -->
    <input type="hidden" name="CLIENTADDRESS" value="<?php echo
$CLIENTADDRESS ?>" />
    <input type="hidden" name="CLIENTEMAIL" value="<?php echo $user['email']
?>" />

    <input type="submit" name="ok" value="Buy with Be2bill " />
</form>
...
```

Paiement PayPal ou Buyster

```php
<?php

define(BE2BILL_IDENTIFIER, 'my_id');
define(BE2BILL_PASSWORD, 'this_is_a_secret');

$AMOUNT = intval($order['amount'] * 100);
$CLIENTIDENT = $user['id'] . '_' . $user['login'];

$hash = be2bill_signature(…) ;

#Initilize the first (registering) request params
$params = array(
    'method' => 'payment',
    'params' => array
    (
        'IDENTIFIER'        => BE2BILL_IDENTIFIER,
        'HASH'              => $hash,
        'AMOUNT'            => $AMOUNT,
        'CLIENTIDENT'       => $CLIENTIDENT,
        'CLIENTREFERRER'    => $_SERVER['HTTP_REFERER'],
        'CLIENTUSERAGENT'   => $_SERVER['HTTP_USER_AGENT'],
        'CLIENTIP'          => $_SERVER['REMOTE_ADDR'],
        'CLIENTEMAIL'       => $user['email'],
        'DESCRIPTION'       => $order['description'],
        'ORDERID'           => $order['id'],
        'CARDFULLNAME'      => $_POST['fullname'],
        'VERSION'           => '2.0',
    )
);

#Prepare HTTPS request with curl
$resource = curl_init();

curl_setopt($resource, CURLOPT_URL,
'https://$site/front/service/rest/process');
curl_setopt($resource, CURLOPT_RETURNTRANSFER, true);
curl_setopt($resource, CURLOPT_POST, true);
curl_setopt($resource, CURLOPT_POSTFIELDS, http_build_query($params));
#Send the request
$serialized = curl_exec($resource);

#Parse result
$result = json_decode($serialized, true);

if $result['EXECCODE'] == '0002')
{
```

```
    echo base64_decode( $result['REDIRECTHTML'] );
    exit;
}
else
{
    #TODO: error handling
}

curl_close($resource);
```

Paiement 1 clic serveur à serveur

```php
<?php

define(BE2BILL_IDENTIFIER, 'my_id');
define(BE2BILL_PASSWORD, 'this_is_a_secret');

$AMOUNT = intval($order['amount'] * 100);
$CLIENTIDENT = $user['id'] . '_' . $user['login'];
$CLIENTADDRESS = $user['address1'] . $user['address2'] . $user['cp'] .
$user['city'] . $user['country'];

$hash = be2bill_signature(…) ;

#Initilize the first (registering) request params
$params = array(
    'method' => 'payment',
    'params' => array
    (
        'IDENTIFIER'          => BE2BILL_IDENTIFIER,
        'HASH'                => $hash,
        'AMOUNT'              => $AMOUNT,
        'CLIENTIDENT'         => $CLIENTIDENT,
        'CLIENTREFERRER'      => $_SERVER['HTTP_REFERER'],
        'CLIENTUSERAGENT'     => $_SERVER['HTTP_USER_AGENT'],
        'CLIENTIP'            => $_SERVER['REMOTE_ADDR'],
        'CLIENTEMAIL'         => $user['email'],
        'DESCRIPTION'         => $order['description'],
        'ORDERID'             => $order['id'],
        'LANGUAGE'            => $user['language'],
        'CARDFULLNAME'        => $_POST['fullname'],
        'CARDCODE'            => $_POST['cardcode'],
        'CARDVALIDITYDATE'    => $_POST['valdate'],
        'CARDCVV'             => $_POST['cvv'],
        'CREATEALIAS'         => 'yes',
        'VERSION'             => '2.0',
    )
);

#Prepare HTTPS request with curl
$resource = curl_init();

curl_setopt($resource, CURLOPT_URL,
'https://$site/front/service/rest/process');
curl_setopt($resource, CURLOPT_RETURNTRANSFER, true);
curl_setopt($resource, CURLOPT_POST, true);
curl_setopt($resource, CURLOPT_POSTFIELDS, http_build_query($params));
```

```php
#Send the request
$serialized = curl_exec($resource);

#Parse result
$result = json_decode($serialized, true);

if ($result['EXECCODE'] == 0)
{
    #Transaction succeed
    echo $result['TRANSACTIONID'];

    #Save ALIAS in database
    $database->query('UPDATE user SET user.alias = ' . $result['ALIAS'] . '
WHERE user.id = ' . $user['id'];
}
else
{
    #TODO: error handling
}

curl_close($resource);

/* Few days later we want to process another payment without asking for card
details to holder */

...
$hash = be2bill_signature(…) ;

#Initilize an aliased request params
$params = array(
    'method' => 'payment',
    'params' => array
    (
        'IDENTIFIER'         => BE2BILL_IDENTIFIER,
        'HASH'               => $hash,
        'AMOUNT'             => $AMOUNT,
        'CLIENTIDENT'        => $CLIENTIDENT,
        'CLIENTREFERRER'     => $_SERVER['HTTP_REFERER'],
        'CLIENTUSERAGENT'    => $_SERVER['HTTP_USER_AGENT'],
        'CLIENTIP'           => $_SERVER['REMOTE_ADDR'],
        'CLIENTEMAIL'        => $user['email'],
        'DESCRIPTION'        => $order['description'],
        'ORDERID'            => $order['id'],
        'LANGUAGE'           => $user['language'],
        'ALIAS'              => $user['alias'],
        'ALIASMODE'          => 'oneclick',
        'VERSION'            => '2.0',
    )
);
```

```php
#Prepare HTTPS request with curl
$resource = curl_init();

curl_setopt($resource, CURLOPT_URL,
'https://$site/front/service/rest/process');
curl_setopt($resource, CURLOPT_RETURNTRANSFER, true);
curl_setopt($resource, CURLOPT_POST, true);
curl_setopt($resource, CURLOPT_POSTFIELDS, http_build_query($params));

#Send the request
$serialized = curl_exec($resource);

#Parse result
$result = json_decode($serialized, true);

if ($result['EXECCODE'] == 0)
{
    #Transaction succeed
    echo $result['TRANSACTIONID'];
}
else
{
    #TODO: error handling
}
curl_close($resource);
```

Autorisation et capture serveur à serveur

```php
<?php

define(BE2BILL_IDENTIFIER, 'my_id');
define(BE2BILL_PASSWORD, 'this_is_a_secret');

$AMOUNT = intval($order['amount'] * 100);
$CLIENTIDENT = $user['id'] . '_' . $user['login'];
$CLIENTADDRESS = $user['address1'] . $user['address2'] . $user['cp'] .
$user['city'] . $user['country'];

$hash = be2bill_signature(…) ;

#Initilize authorization request params
$params = array(
    'method' => 'authorization',
    'params' => array
    (
        'IDENTIFIER'          => BE2BILL_IDENTIFIER,
        'HASH'                => $hash,
        'AMOUNT'              => $AMOUNT,
        'CLIENTIDENT'         => $CLIENTIDENT,
        'CLIENTREFERRER'      => $_SERVER['HTTP_REFERER'],
        'CLIENTUSERAGENT'     => $_SERVER['HTTP_USER_AGENT'],
        'CLIENTIP'            => $_SERVER['REMOTE_ADDR'],
        'CLIENTEMAIL'         => $user['email'],
        'DESCRIPTION'         => $order['description'],
        'ORDERID'             => $order['id'],
        'LANGUAGE'            => $user['language'],
        'CARDFULLNAME'        => $_POST['fullname'],
        'CARDCODE'            => $_POST['cardcode'],
        'CARDVALIDITYDATE'    => $_POST['valdate'],
        'CARDCVV'             => $_POST['cvv'],
        'VERSION'             => '2.0',
    )
);

#Prepare HTTPS request with curl
$resource = curl_init();

curl_setopt($resource, CURLOPT_URL,
'https://$site/front/service/rest/process');
curl_setopt($resource, CURLOPT_RETURNTRANSFER, true);
curl_setopt($resource, CURLOPT_POST, true);
curl_setopt($resource, CURLOPT_POSTFIELDS, http_build_query($params));
```

```
#Send the request
$serialized = curl_exec($resource);

#Parse result
$result = json_decode($serialized, true);

if ($result['EXECCODE'] == 0)
{
    #Transaction succeed
    echo $result['TRANSACTIONID'];

    #Save TRANSACTIONID in database
    $database->query('UPDATE order SET succeed_authorization_id = ' .
$result['TRANSACTIONID'] . ' WHERE order.id = ' . $order['id'];
}
else
{
    #TODO: error handling
}

curl_close($resource);

/* Few days later we want to capture the funds from the authorization
transaction */

...

$hash = be2bill_signature(…) ;

#Initilize authorization request params
$params = array(
    'method' => 'capture',
    'params' => array
    (
        'IDENTIFIER'        => BE2BILL_IDENTIFIER,
        'HASH'              => $hash,
        'TRANSACTIONID'     => $order['succeed_authorization_id'],
        'DESCRIPTION'       => $order['description'],
        'ORDERID'           => $order['id'],
        'VERSION'           => '2.0',
    )
);

#Prepare HTTPS request with curl
$resource = curl_init();

curl_setopt($resource,                                                  CURLOPT_URL,
'https://$site/front/service/rest/process');
curl_setopt($resource, CURLOPT_RETURNTRANSFER, true);
curl_setopt($resource, CURLOPT_POST, true);
```

```
curl_setopt($resource, CURLOPT_POSTFIELDS, http_build_query($params));

#Send the request
$serialized = curl_exec($resource);

#Parse result
$result = json_decode($serialized, true);

if ($result['EXECCODE'] == 0)
{
    #Transaction succeed
    echo $result['TRANSACTIONID'];
}
else
{
    #TODO: error handling
}

curl_close($resource);
```

Signature HASH avec be2bill_signature

```php
<?php

/*
 * Create the HASH from Be2bill parameters
 *
 * @param $password The account password
 * @param $parameters A Be2bill (and only Be2bill) parameters array
 *
 * @return The HASH signature
 */
function be2bill_signature($password, array $parameters)
{
    #Alpha sort
    ksort($parameters);

    $clear_string = $password;
    foreach ($parameters as $key => $value)
    {
        if (is_array($value) == true)
        {
            ksort($value);
            foreach ($value as $index => $val)
            {
                $clear_string .= $key . '[' . $index . ']=' . $val .
$password;
            }
        }
        else
        {
            $clear_string .= $key . '=' . $value . $password;
        }
    }

    return hash( 'sha256' , $clear_string );
}
?>
```

A propos de PHP4

SHA256

En langage PHP4 vous devrez utiliser la libraire mhash qui permet d'utiliser l'algorithme SHA256.

Exemple :

```
bin2hex(mhash(MHASH_SHA256, 'foo')) == hash('sha256', 'foo');
```

json_decode

En langage PHP4 vous devez utiliser des librairies implémentant le format JSON

Exemple :

- http://pear.php.net/package/Services_JSON
- http://pecl.php.net/package/json